Exhibit 2

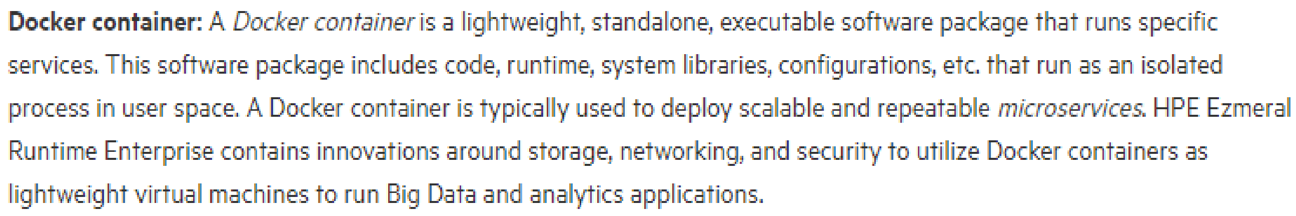
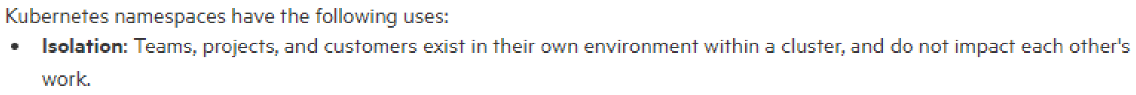
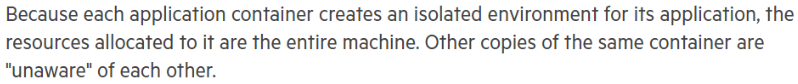
U.S. Patent No. 7,519,814 vs. HPE

Accused Instrumentalities: HPE products and services using secure containerized applications, including without limitation HPE's Ezmeral Runtime Enterprise and HPE GreenLake, and all versions and variations thereof since the issuance of the asserted patent.

Each Accused Instrumentality infringes the claims in substantially the same way, and the evidence shown in this chart is similarly applicable to each Accused Instrumentality. Each claim limitation is literally infringed by each Accused Instrumentality. However, to the extent any claim limitation is not met literally, it is nonetheless met under the doctrine of equivalents because the differences between the claim limitation and each Accused Instrumentality would be insubstantial, and each Accused Instrumentality performs substantially the same function, in substantially the same way, to achieve the same result as the claimed invention. Notably, Defendant has not yet articulated which, if any, particular claim limitations it believes are not met by the Accused Instrumentalities.

Claim 1

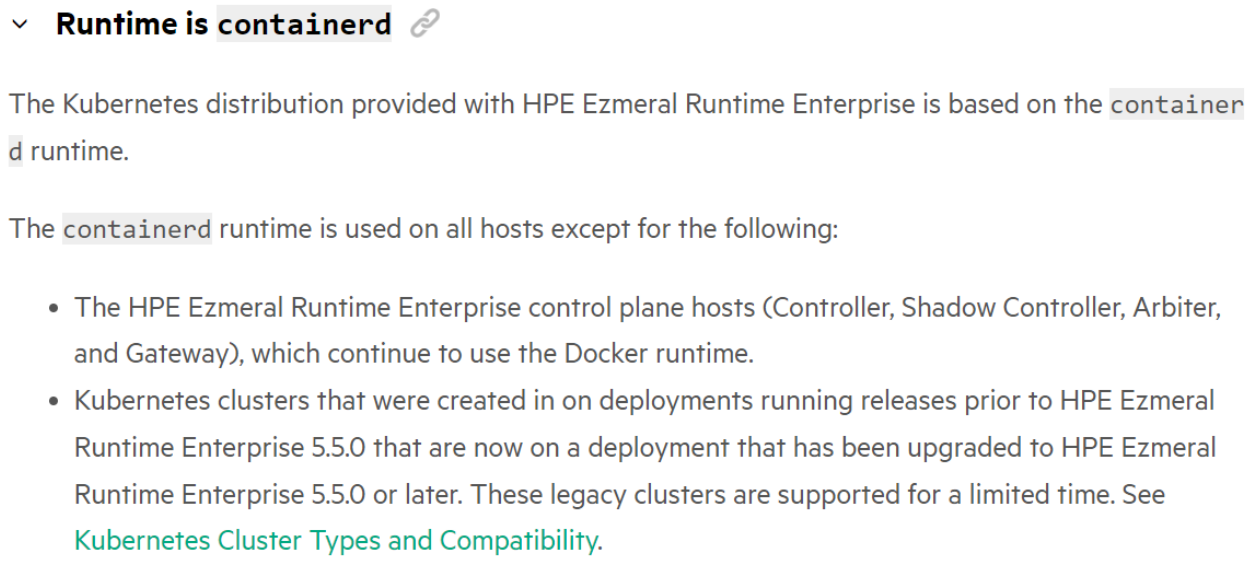
Claim 1	Accused Instrumentalities
[1pre] 1. In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor, a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications are executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising:	To the extent the preamble is limiting, HPE and/or its customer practices, through the Accused Instrumentalities, in a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor, a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications are executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service, as claimed. For example, HPE Ezmeral Runtime Enterprise runs on individual servers, including HPE Synergy and HPE ProLiant servers, each of which runs an independent operating system, including for example RHEL or SLES running either on bare metal, through an on-premises virtualized infrastructure, through one or more cloud services, or through any other supported deployment. In an exemplary deployment, two or more servers use different operating systems. HPE requires that each server includes a processor with one or more cores available to the OS kernel. HPE further requires each server to have a supported operating system (SLES or RHEL/CentOS), which includes a kernel and associated local system files, including for example libraries such as libc/glibc, configuration files, etc. In the infringing system, at least two servers have different

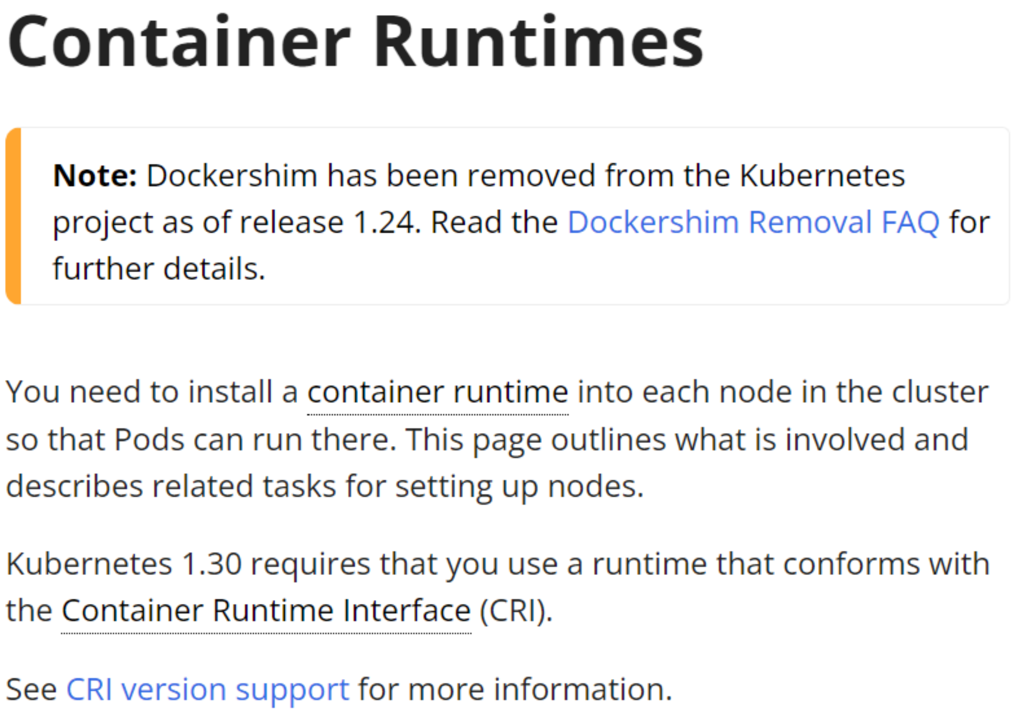
Claim 10	Accused Instrumentalities
to system files within the operating system during execution thereof.	*See, e.g.*:  **Docker container:** A *Docker container* is a lightweight, standalone, executable software package that runs specific services. This software package includes code, runtime, system libraries, configurations, etc. that run as an isolated process in user space. A Docker container is typically used to deploy scalable and repeatable *microservices*. HPE Ezmeral Runtime Enterprise contains innovations around storage, networking, and security to utilize Docker containers as lightweight virtual machines to run Big Data and analytics applications. https://support.hpe.com/hpesc/public/docDisplay?docId=a00ecp54hen_us&docLocale=en_US&page=home/about-hpe-ezmeral-container-pl/GEN_Definitions.html  Kubernetes namespaces have the following uses: - **Isolation:** Teams, projects, and customers exist in their own environment within a cluster, and do not impact each other's work. https://support.hpe.com/hpesc/public/docDisplay?docId=a00ecp55hen_us&docLocale=en_US&page=reference/universal-concepts/Namespaces.html  Because each application container creates an isolated environment for its application, the resources allocated to it are the entire machine. Other copies of the same container are "unaware" of each other. https://developer.hpe.com/blog/kubernetes-application-containers-managing-containers-and-cluster-resour/

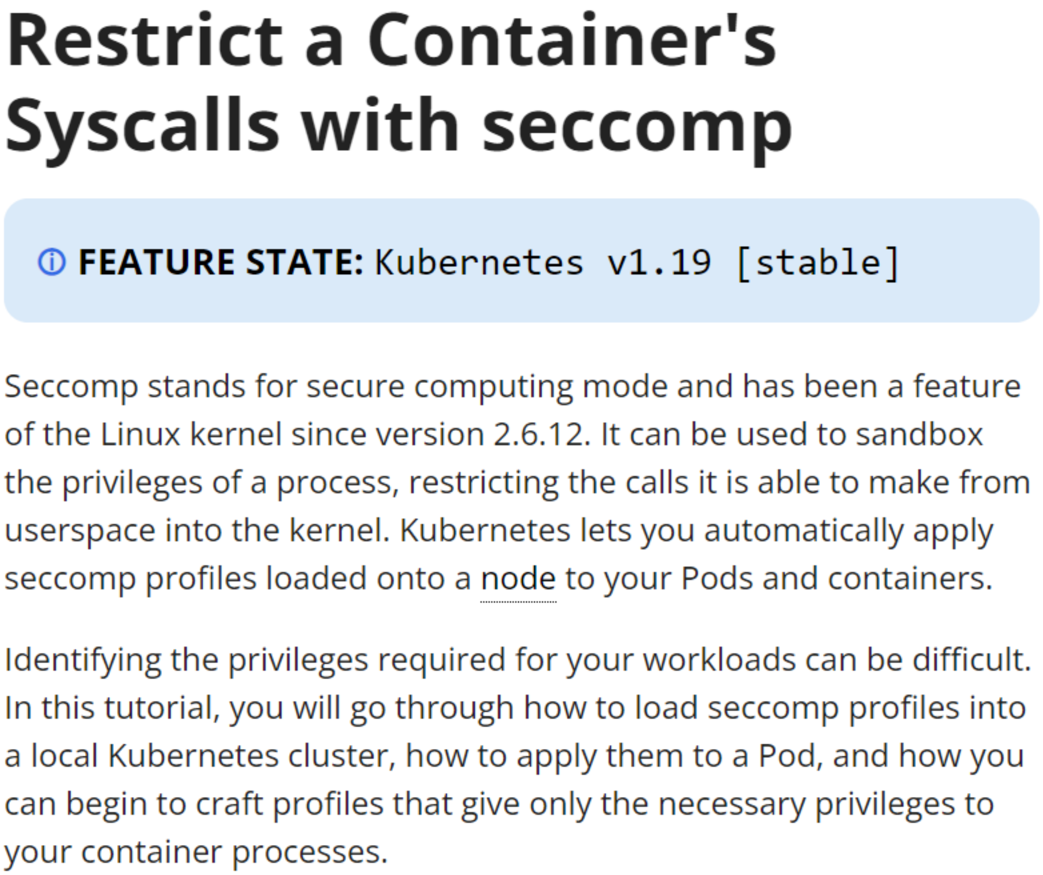
Claim 31

Claim 31	Accused Instrumentalities
[31pre] A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:	To the extent the preamble is construed as a limitation, each Accused Instrumentality is or comprises a computing system for performing a plurality of tasks each comprising a plurality of processes. *See* claim limitations below. *See also* analysis and evidence for [1pre] above.
[31a] a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers, each container being mutually exclusive of the other, such that read/write files within a container cannot	Each Accused Instrumentality comprises a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers, each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers, each container of files is said to have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a Mac_address.

Claim 31	Accused Instrumentalities
be shared with other containers, each container of files is said to have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a Mac_address	*See* analysis and evidence for [1pre], limitations [1a] and [1f], and claim 6 above.
[31b] wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes wherein the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system, each container having its own execution file associated therewith for starting one or more applications, in operation, each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel; and,	Each Accused Instrumentality comprises a system wherein the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes wherein the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system, each container having its own execution file associated therewith for starting one or more applications, in operation, each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel. *See* analysis and evidence for [1pre], limitations [1a], [1c], [1d], [1e], and [1f], and claim 2 above.
[31c] a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.	Each Accused Instrumentality comprises a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications. For example, HPE Ezmeral Runtime Enterprise includes either the Docker or containerd runtime module. For another example, Kubernetes uses the Linux kernel's seccomp mode to monitor and control system calls made from a container. *See, e.g.*:

Claim 31	Accused Instrumentalities
	 **Runtime is `containerd`** The Kubernetes distribution provided with HPE Ezmeral Runtime Enterprise is based on the `containerd` runtime. The `containerd` runtime is used on all hosts except for the following: • The HPE Ezmeral Runtime Enterprise control plane hosts (Controller, Shadow Controller, Arbiter, and Gateway), which continue to use the Docker runtime. • Kubernetes clusters that were created in on deployments running releases prior to HPE Ezmeral Runtime Enterprise 5.5.0 that are now on a deployment that has been upgraded to HPE Ezmeral Runtime Enterprise 5.5.0 or later. These legacy clusters are supported for a limited time. See Kubernetes Cluster Types and Compatibility. https://docs.ezmeral.hpe.com/runtime-enterprise/56/reference/kubernetes/hewlett_packard_enterprise_distributions_of_kubernetes.html

Claim 31	Accused Instrumentalities
	 Container Runtimes **Note:** Dockershim has been removed from the Kubernetes project as of release 1.24. Read the Dockershim Removal FAQ for further details. You need to install a container runtime into each node in the cluster so that Pods can run there. This page outlines what is involved and describes related tasks for setting up nodes. Kubernetes 1.30 requires that you use a runtime that conforms with the Container Runtime Interface (CRI). See CRI version support for more information. https://kubernetes.io/docs/setup/production-environment/container-runtimes/

Claim 31	Accused Instrumentalities
	 **Restrict a Container's Syscalls with seccomp** ⓘ **FEATURE STATE:** `Kubernetes v1.19 [stable]` Seccomp stands for secure computing mode and has been a feature of the Linux kernel since version 2.6.12. It can be used to sandbox the privileges of a process, restricting the calls it is able to make from userspace into the kernel. Kubernetes lets you automatically apply seccomp profiles loaded onto a node to your Pods and containers. Identifying the privileges required for your workloads can be difficult. In this tutorial, you will go through how to load seccomp profiles into a local Kubernetes cluster, how to apply them to a Pod, and how you can begin to craft profiles that give only the necessary privileges to your container processes. https://kubernetes.io/docs/tutorials/security/seccomp/